



Components

- QSRV
 - PVA server access to process database (aka. records)
- PVA links https://epics-base.github.io/pva2pva/qsvr_page.html
 - PVA client access from process database
- P4P
 - Python bindings for PVA client
 - Server bindings in progress <https://mdavidsaver.github.io/p4p/>
- P2P
 - PVA 2 PVA
 - gateway/proxy <https://github.com/epics-base/pva2pva/blob/master/loopback.conf>
- “simple” (but not EZ) c++ client API
 - <http://mdavidsaver.github.io/pvAccessCPP/>



QSRV

- QSRV \geq RSRV
- Remote access to PVs
 - As you've always known them
 - “Single PVs”
 - No additional configuration
- Access to Group PVs
 - Groups of single PVs
 - Accessed atomically
 - Additional configuration required



Features

- Done
 - Supports PVA get, put, and monitor
 - Single + Group
 - Put w/ callback
 - Single only
- Future
 - Access Security

Group PV Definition

```
record(ai, "rec:X") {  
    info(Q:group, {  
        "grp:name": {  
            "X": {+channel:"VAL"}  
        }  
    })  
}  
record(ai, "rec:Y") {  
    info(Q:group, {  
        "grp:name": {  
            "Y": {+channel:"VAL"}  
        }  
    })  
}
```

```
$ pvget grp:name  
grp:name  
structure  
    epics:nt/NTScalar:1.0 X  
        double value 0  
        alarm_t alarm INVALID DRIVER UDF  
        time_t timeStamp <undefined> 0  
    ...  
    epics:nt/NTScalar:1.0 Y  
        double value 0  
        alarm_t alarm INVALID DRIVER UDF  
        time_t timeStamp <undefined> 0  
    ...
```

Group Definitions

- Documentation
 - https://epics-base.github.io/pva2pva/qsvr_page.html
- Examples
 - <https://github.com/epics-base/pva2pva/tree/master/iocBoot>
 - iocwf demo
 - ~2 channel ADC (I and Q)
 - iocimagedemo
 - NTNDArray w/ faked data (not AD)
- Caveats
 - Group Monitor \approx Triggered Get
 - Doesn't use dbEvent queue
 - No Group put w/ callback



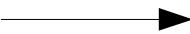
PVA Links

- Documentation
 - https://epics-base.github.io/pva2pva/qsvr_page.html#qsvr_link
- Example
 - <https://github.com/epics-base/pva2pva/tree/master/iocBoot/iocPvalink>



PVA Links (2)

```
record(longin, "tgt") {}  
record(longin, "src") {  
    field(INP, {pva:"tgt"})  
}
```



```
record(longin, "tgt") {}  
record(longin, "src") {  
    field(INP, {pva:{pv:"tgt"}})  
}
```

- Use JSON syntax
- INP/OUT/FLNK
- Lots of options
- “Local” links
 - Direct to QSRV, no network



PVA Links (3)

```
record(longin, "tgt") {}  
record(longin, "src") {  
    field(INP, {pva:{  
        pv:"tgt",  
        field:"",  
        local:false,  
        Q:4,  
        pipeline:false,  
        proc:none,  
        sevr:false,  
        time:false,  
        monorder:0,  
        retry:false,  
        always:false,  
        defer:false  
    }})  
}
```

- Target PV name
 - Can include Server side plugins
 - eg. “blah.VAL[:4]”
- none, false/“NPP”, true/“PP”, “CP”, “CPP”
 - Processing. Applies to put/monitor
- false, true, “MSI”
 - Maximize severity
- false/true
 - Update .TIME

PVA Links (4)

- Operate on sub-structure
 - Require local PV
 - Monitor Q depth
 - Enable Monitor flow control
 - Order of processing during CP scan
 - Queue put while disconnected
 - Proc on-change or always
 - Always Queue put
- ```
record(longin, "tgt") {}
record(longin, "src") {
 field(INP, {pva:{
 pv:"tgt",
 field:"",
 local:false,
 Q:4,
 pipeline:false,
 proc:none,
 sevr:false,
 time:false,
 monorder:0,
 retry:false,
 always:false,
 defer:false
 }})
}
```

